Single-pass Incremental Force Updates for Adaptively Restrained Molecular Dynamics

Krishna Kant Singh, Stephane Redon

Single-pass Incremental Force Updates for Adaptively Restrained Molecular Dynamics

Krishna Kant Singh and Stephane Redon *

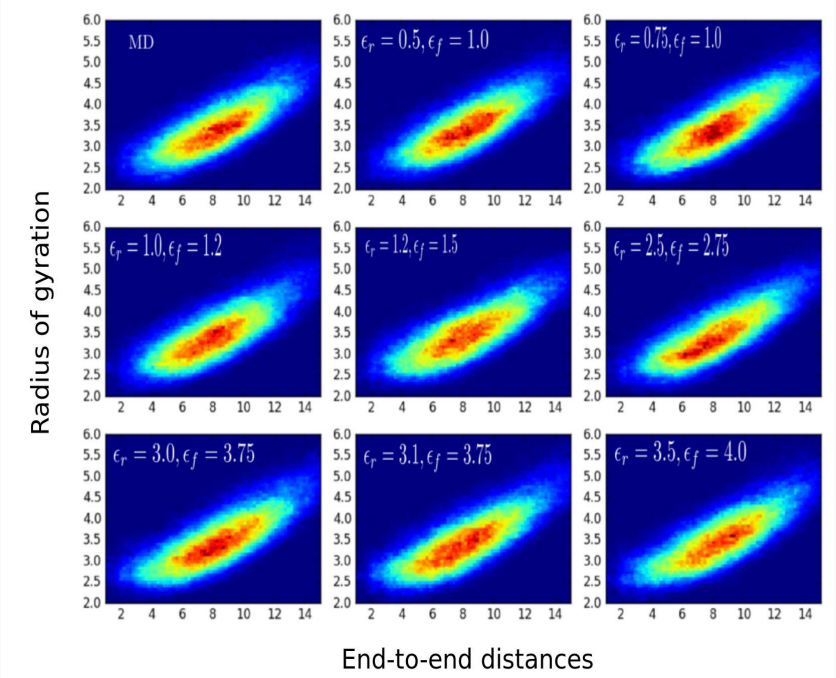
November 11, 2017

Abstract

Adaptively Restrained Molecular dynamics (ARMD) allows users to perform more integration steps in wall-clock time by switching on and off positional degrees of freedoms. This article presents new, single-pass incremental force updates algorithms to efficiently simulate a system using ARMD. We assessed different algorithms for speedup measurements and implemented them in the LAMMPS MD package. We validated the *single-pass incremental force update* algorithm on four different benchmarks using diverse pair potentials. The proposed algorithm allows us to perform simulation of a system faster than traditional MD in both NVE and NVT ensembles. Moreover, ARMD using the new single-pass algorithm speeds up the convergence of observables in wall-clock time.

Keywords: Neighbor Lists, Incremental Force Update, Adaptively Restrained Molecular Dynamics.

*NANO-D, INRIA; Univ. Grenoble Alpes, LJK, F-38000 Grenoble, France; CNRS, LJK, F-38000 Grenoble, France



The figure represents the configuration space explore by all-atom MD and ARMD (with different restraining parameters) simulations. Adaptively switching the positional degrees of freedom on and off not only allows to perform more number of integration steps in wall clock time but also produces the unbiased and equilibrium statistics of the system.

1 Introduction

Molecular Dynamics (MD) may serve as a computational microscope to decipher the structural and dynamical behavior of complex systems[1]. In particular, MD may provide atomistic descriptions over several time scales (*e.g.* femtoseconds to milliseconds)[2–7]. Unfortunately, simulating complex systems over experimental time scales is still computationally challenging[8].

Numerous attempts have been made to either speed up simulations or enhance phase-space sampling, for example through the use of additional biased potentials, which help trajectories escape from local minima[8–11]. Complex systems can also be simplified through coarse-grained simulations[12–15]. Other strategies involve modifications of the integration time step, which may be increased either by increasing the mass associated with the fastest mode present in the system, by repartitioning the mass, or by creating virtual sites[16,17]. MD simulations have also been accelerated by optimized algorithms that perform mathematical operations faster, sometimes on special-purpose supercomputers[18].

In many simulations, most of the time is spent computing non-bonded forces, typically divided into long-range and short-range forces. Long-range interactions involve electrostatic interactions that are generally computed using specialized methods, including the reaction field method[19,20], the cell-multipole method, the fast multipole method[21], particle Mesh Ewald (PME)[22] or the Wolf method[23–28]. Short-range forces decay fast and are generally truncated at a cut-off distance r_c. These forces are efficiently computed using neighbor lists. With the Wolf method, long-range forces may also be computed using neighbor lists, and this method recently attracted a great deal of attention for computing properties of electrolytes and simulating biomolecules in explicit solvent[25].

Adaptively Restrained Molecular Dynamics (ARMD) is a recent approach that attempts to perform more integration steps per wall-clock second by reducing the number of computations per time step[29]. In ARMD, positional degrees of freedom are adaptively turned on and off during the simulation, based on their instantaneous kinetic energy. When two particles are restrained, the forces they apply on each other do not need to be updated from one time to the next. As a result, *incremental force update algorithms*, *i.e.* force calculations

that keep track of the total force applied on each particle and update forces involving active particles, may significantly speedup the simulation[29–32].

Up to now, however, such incremental algorithms have required two passes: one to subtract interactions involving from the previous time step, and one to add updated interactions for the current time step. As a result, previous incremental force update algorithms have been useful in situations where at least half of the simulated particles were restrained. In this article, we present a *single-pass incremental force update algorithm* for ARMD that overcomes this limitation. The proposed algorithms are implemented in the LAMMPS MD package and validated by performing simulations of diverse systems in both the NVE and NVT ensembles.

2 Adaptively Restrained Molecular Dynamics

For completeness, we briefly review the ARMD methodology[29]. In ARMD, the time evolution of a system containing N particles is obtained by the Adaptively Restrained (AR) Hamiltonian:

$$H_{AR}(\mathbf{q},\mathbf{p}) = \frac{1}{2}\mathbf{p}^T\Phi(\mathbf{q},\mathbf{p})\mathbf{p} + V(\mathbf{q}), \tag{1}$$

where $\Phi(\mathbf{q},\mathbf{p})$ is the inverse inertia matrix that adaptively imposes restraints during the simulation. Precisely, particles whose kinetic energy is smaller that a restrained-dynamics threshold (ε_i^r) are *restrained*, and their positions remain unchanged. Particles whose kinetic energy is greater than a full-dynamics threshold ε_i^f are considered as *active*, and have normal dynamics. Transition particles have kinetic energies between ε_i^r and ε_i^f (in the context of force calculations, transitioning particles are mobile, and are thus treated as, and called, active particles). The status of each particle can evolve during the simulation, and the Adaptively Restrained Hamiltonian (1) ensures that stable simulations can be performed. Equilibrium statistics can be recovered when performing AR Langevin dynamics[33]. In order

to compute position-dependent equilibrium averages, equation 1 can be re-written as:

$$H_{AR} = \frac{1}{2}\mathbf{p}^T M^{-1}\mathbf{p} + V(\mathbf{q}) + \frac{1}{2}\mathbf{p}^T(\phi(\mathbf{q},\mathbf{p}) - M^{-1})\mathbf{p}$$
$$H_{AR} = H + V_{AR}. \tag{2}$$

In equation 2, V_{AR} is a biased term that depends upon the momenta of particles and averages can be estimated[29]. In particular, correct position-dependent equilibrium averages can be estimated as[29]:

$$< A(q) >_{H_{AR}} = \frac{\int A(\mathbf{q}) e^{\frac{-H(\mathbf{q},\mathbf{p})}{KT}} d\mathbf{q}}{\int e^{\frac{-H(\mathbf{q},\mathbf{p})}{KT}} d\mathbf{q}} =< A(q) >_H . \tag{3}$$

From equation 3, it is to be noted that position-dependent averages from an ARMD and an MD simulations are the same.

Due to the biased term V_{AR}, ARMD might not be directly useful to compute dynamical properties or properties that depend upon the fluctuations of kinetic energies.

3 Algorithms for AR Molecular Dynamics

Let C denote the complete list of particles, A denote the list of active particles, and R denote the list of restrained particles. Let S_A (resp. S_R) denote the list of particles that *switched* to being active (resp. restrained) between the previous and current time step.

In ARMD, a system of N particles can be referred to as a combination of N_A active and N_R $(= N - N_A)$ restrained particles. Thus, at any time-step, interactions between particles can be categorized as *active interactions* (interactions involving at least one active particle) and restrained interactions (interactions between restrained particles only). Precisely, active interactions involve forces between either two active particles (F_{AA}) or one active particle and one restrained particle (F_{AR}), while restrained interactions involve forces between two restrained particles (F_{RR}). Figure 1 shows the three types of interactions on an example.

Most often, forces acting on particles only depend upon their relative positions. At a given time step, relative positions between restrained particles do not change; hence, restrained interactions and associated forces remain unchanged. This eliminates the need to compute

F_{RR} force components involving restrained interactions, and the F_{RR} force components from the previous time step can be cached and reused. However, active particles update their positions, and the associated force components (F_{AA} and F_{AR}) need to be updated at each time step.

3.1 Active neighbor lists

In order to compute forces due to active interactions, we previously introduced Active Neighbor Lists (ANLs) *i.e.* lists of active neighbor particles at a given time-step[32]. The ANLs are constructed from Full Neighbor Lists (FNLs), and the ANL of an active particle i can be described as $ANL(i) = \{\forall j \in FNL(i) : j \in R \cup (j \in A \cap (j < i))\}$. In other words, $ANL(i)$ contains all the restrained neighbors of i, and the active neighbors of i with smaller indices (due to the condition $j < i$).

3.2 Two-pass incremental force update algorithm

In MD, the forces acting on all particles are typically computed once, after updating particles positions. In contrast, in ARMD, only the positions of active particles are updated, and forces are updated based on the new positions of active particles, thereby eliminating the need to compute all forces. In our previous approach, active interactions were incrementally updated using an algorithm that involved two force update steps[29,30,32]:

1. **First pass** or subtraction step: This pass removes the force increments involving active particles based on the old positions, by computing the force increments due to active particles and subtracting them from the total force.

2. Position update step: This involves updating the positions of active particles only, instead of updating the positions of all particles.

3. **Second pass** or addition step: This step involves computing force increments based on the updated positions of active particles, which are then added to the forces obtained from the subtraction step.

This incremental force update algorithm requires force increments to be calculated twice in a single integration step, limiting the usefulness of ARMD to cases where at least 50% of particles are restrained[32]. Furthermore, if all particles are active, the proposed incremental force algorithm would be twice times slower than the normal MD algorithm.

3.3 Single-pass incremental force updates

In order to overcome the aforementioned limitation, we have designed a *single-pass* incremental force computation algorithm.

In ARMD, the force F_i acting on a particle i can be written as $F_i = F_{i_{AA}} + F_{i_{AR}} + F_{i_{RR}}$ (figure 1), where $F_{i_{AA}}$ refers to forces between two active particles, $F_{i_{AR}}$ refers to forces between an active and a restrained particle and $F_{i_{RR}}$ refers to the force between two restrained particles.

Assume there are no switching particles between the n-th and $(n+1)$-th time step. Using the two-pass incremental force computation algorithm, the position of an active particle i is updated, and the total force $F_i = F_{i_{AA}} + F_{i_{AR}}$ applied to it is updated during the addition step. Since the new position is not updated *after* the addition step, though, the next substraction step substracts the *same* force increments, yielding $F_i = 0$ after the subtraction step. Therefore, in case of an active particle, the subtraction step can be replaced by assigning zero to the associated force.

In case of a restrained particle i, the total force applied to it is $F_i = F_{i_{AR}} + F_{i_{RR}}$. As noted above, $F_{i_{RR}}$ can be cached and reused, so that we only have to compute $F_{i_{AR}}$. In single-pass algorithm, force on the particle i is assigned as $F_i = F_{i_{RR}}$ before computing the force component, unlike the previous algorithm that adds and removes $F_{i_{AR}}$ force component (as the $F_{i_{RR}}$ force component remains unchanged and could be retained from the previous time step). Also, this algorithm eliminates the need to carry out subtraction step and subtraction step can be removed with algorithm 1. However, the time evolution of the system switches the state of the particles, force components associated with these switched particles need to be computed, before applying algorithm 1. Particles can switch their states in two ways:

1. Switching to active state

2. Switching to restrained state

It is likely that the switching in the state of particles may cause the change in the type of interaction (to an active or restrained interaction) and hence the associated force components.

An active interaction associated with pair $i-j$ can switch to a restrained interaction in the following two ways:

- Assuming at time step n both particles i and j are active and they switch to a restrained state at $n+1$ time step, the F_{AA} force component associated with the pair $i-j$ would now switch to the F_{RR} force component. As ANLs avoid computation of F_{RR} force component, this switched force component needs to be computed and stored for next time steps.

- On the other hand, if we assume particles i and j to be active and restrained respectively at a time step n (associated force component has type F_{AR}), and at $n+1$ time step the particle i changes to a restrained state, this switching updates the interaction type from active to restrained and associated force component from F_{AR} to F_{RR}. As mentioned previously, the updated F_{RR} force component needs to be computed and stored for the next time-steps.

Similarly, a restrained interaction belonging to a pair $i-j$ can switch to an active interaction in the following ways:

- In case, particle i or j switches their state from restrained to an active state, force component F_{RR} associated with this pair switches to F_{AR}. This updated F_{AR} force component needs to be computed and subtracted from the forces of restrained neighbors of the this switched particle i.

- Another possibility wherein both particles i and j switch from active to restrained state, the associated F_{RR} force component switches to the F_{AA} force components. This type of switching does not require any force computation as zero can be assigned for the forces associated with i and j.

In order to compute the switched force components (F_{RR} or F_{AR}), we used a modified active neighbor list ANL'. The ANL' of a switched particle i is an extracted ANL that contains only restrained neighbors. Algorithm 2 gives a description of extracting the ANL' from the ANL. Algorithm 3 shows the pseudo-code to compute switched force components. Line 3 of algorithm 3 computes F_{RR} force components and stores them in F^+ and line 9 computes F_{AR} force components and stores then in F^- for the switched particles.

Algorithm 1 gives the pseudo-code to perform the ARMD integration step using the ANLs and incremental force update algorithm. In this algorithm, before computing forces based on the new positions, zeros (line 3) and the force components F_{RR} (line 6) are assigned to the forces corresponding to active and restrained particles respectively.

Algorithm 1: $SinglePassIncrementalForceComputation()$

```
1  for i ∈ C do
2      if i ∈ A then
3          F_i^+ ← 0
4      end
5      else
6          F_i^+ ← F_i^+ − F_i^−
7      end
8      F_i^− ← 0
9  end
10 F ← F^+ + ComputeForces(A, ANL)
```

4 Analysis

In this section, we assess different algorithms for constructing ANLs and incrementally update forces. We compare these algorithms with brute-force MD algorithms which have a running time equal to $\tau_{MD} = N * \tau_{FNL} + N * \tau_F$, where N is the total number of particles, τ_{FNL} is the time needed to build the FNL of one particle, and τ_F is the time required to

Algorithm 2: $ExtractANL'(i)$

$ANL'(i) \leftarrow \emptyset$
for $j \in ANL(i)$ **do**
    if $(i \in R \; and \; j \in R) \; or (i \in A \; and \; (j \in R \; and \; j \notin S))$ **then**
        $ANL'(i) \leftarrow ANL'(i) \cup j$
    end
end

Algorithm 3: $SwitchedForces()$

for $i \in S_R$ **do**
    $ExtractANL'(i)$
    $F^+ \leftarrow F^+ + ComputeForces(i, ANL'(i))$
    $ClearANL(i)$
end
for $i \in S_A$ **do**
    $BuildANL(i)$
    $ExtractANL'(i)$
    $F^- \leftarrow F^- + ComputeForces(i, ANL'(i))$
end

Algorithm 4: ARMD integration step

```
1  Update momenta
2  Update A, R, S_A, S_R
3  if (UpdateNeeded) then
4      for i ∈ C do
5          if i ∈ A then
6              Build FNL(i) and ANL(i) simultaneously
7          end
8          else
9              BuildFNL(i)
10         end
11     end
12 end
13 else
14     SwitchedForces()
15 end
16 Update Positions
17 SinglePassIncrementalForceComputation()
```

compute the total force acting on one particle due to its neighbors. To simplify the analysis, we consider the timing required to construct the FNL rather than the HNL.

4.1 Time complexity

In this algorithm, the FNLs of all particles and the ANLs of active particles were constructed at the same time. The ANLs of particles that become active were constructed using the FNL of the corresponding particle, as building the ANLs from scratch is relatively more time consuming. The force components F_{AR} or F_{RR} for switched particles were then computed in two steps (Algorithm 3). The first step involves extracting ANL' from ANL, and the second step involves computing specific force components with ANL'.

The computation time for algorithm 4 is given by:

$$\tau_{ARMD1} = N * \tau_{FNL} + N_A * \tau_F + N_{SA} * \tau_{FANL} + N_{SA} * \tau_{SF} + N_{SR} * (\tau_{SF} + \tau_{CL}) + N_S * \tau_{Ex} \quad (4)$$

τ_{Ex} : Time to build the ANL' for one particle.
τ_{FANL} : Time to construct the ANL from the FNL.
τ_{SF} : Time to compute specific force components (F_{AR} or F_{RR}).
τ_{CL} : Time to clear an ANL.
N_{SA} : Number of particles switching to an active state.
N_{SR} : Number of particles switching to a restrained state.
N_S : Total number of switched particles ($N_{SA} + N_{SR}$).
Note that Equation (4) does not include τ_{ANL}, the time to compute an ANL, since, when neighbor lists are update from scratch (lines 3-12), Algorithm 4 computes FNLs and ANLs simultaneously.

Algorithm 4 is more efficient than classical MD when $\tau_{ARMD1} < \tau_{MD}$, *i.e.*:

$$N_A * \tau_F + N_{SA} * \tau_{FANL} + N_{SA} * \tau_{SF} + N_{SR} * (\tau_{SF} + \tau_{CL}) + N_S * \tau_{Ex} < N * \tau_F. \quad (5)$$

To simplify the analysis, we assume the following worst cases:

Assumption 1: All switching particles switch to an active state, making it necessary to construct $ANLs$ for all switched particles: $N_{SR} = 0$ and $N_S = N_{SA}$.

Assumption 2: Computing a force component F_{AR} or F_{RR} for a particle takes as much time as computing the total force on this particle: $\tau_{SF} = \tau_F$.

Assumption 3: Even though a) computing the ANL of a particle from its FNL and b) extracting the ANL' of a particle from its ANL are both much more efficient than computing the FNL of a particle, we assume that the sum of τ_{FANL} and τ_{Ex} is equal to τ_{FNL}: $\tau_{FNL} = \tau_{FANL} + \tau_{Ex}$.

Considering these three assumptions, inequality 5 holds when:

$$N_A * \tau_F + N_S * \tau_{FNL} + N_S * \tau_F < N * \tau_F, \tag{6}$$

i.e. when:

$$\frac{N_A}{N} + \frac{N_S}{N}(1 + \frac{\tau_{FNL}}{\tau_F}) < 1. \tag{7}$$

4.2 Optimization

Although Algorithm 4 computes the $ANLs$ and $ANL's$ required to perform the single-pass force update algorithm, it computes the $FNLs$ of all particles when neighbor lists are updated from scratch. Since the FNL of a restrained particle is only useful when it switches to an active state, however, Algorithm 4 may be optimized by maintaining $hasFNL$, a list of particles for which the FNL has been computed, and using this list to determine when to update the various neighbor lists.

Algorithm 5 describes the optimized version. When a particle switches to a restrained state, the ANL of this particle is cleared, while retaining its FNL. When a particle switches to an active state and it does not have a FNL, its ANL and FNL are constructed *on-the-fly* (*i.e.* outside the neighbor list construction step). When a particle switches to an active state and already has an FNL (such as when a particle switches states multiple times between two neighbor list construction steps), the ANL of this particle is constructed from the existing FNL. Algorithm 6 shows how Algorithm 3 is modified in the optimized case, when $hasFNL$ is available.

The computation time for Algorithm 5 is given by:

$$\tau_{ARMD2} = N_A*\tau_{FNL}+N_A*\tau_F+N'_{SA}*\tau_{FNL}+(N_{SA}-N'_{SA})*\tau_{FANL}+N_{SA}*\tau_F+N_{SR}*(\tau_{SF}+\tau_{CL})+N_S*\tau_{Ex}, \tag{8}$$

Algorithm 5: ARMD integration step

```
1  Update momenta
2  Update A, R, S_A, S_R
3  if (UpdateNeeded) then
4      hasFNL ← ∅
5      for i ∈ A do
6          Build FNL(i) and ANL(i) simultaneously
7          hasFNL ← hasFNL ∪ {i}
8      end
9  end
10 else
11     SwitchedForces'()
12 end
13 Update Positions
14 SinglePassIncrementalForceComputation()
```

Algorithm 6: $SwitchedForces'()$

for $i \in S_R$ **do**
    $ExtractANL'(i))$
    $F^+ \leftarrow F^+ + ComputeForces(i, ANL'(i))$
    $ClearANL(i)$
end
for $i \in S_A$ **do**
    if $i \in hasFNL$ **then**
        $BuildANL(i)$
    end
    else
        Build $FNL(i)$ and $ANL(i)$ simultaneously
        $hasFNL \leftarrow hasFNL \cup \{i\}$
    end
    $ExtractANL'(i)$
    $F^- \leftarrow F^- + ComputeForces(i, ANL'(i))$
end

where N'_{SA} is the number of particles without a FNL that have switched to an active state.

In order to compare the time complexity of Algorithm 5 with the MD algorithm, we considered the same assumptions as in the Algorithm 4) ($N_{SR} = 0$, $\tau_{FANL} + \tau_{Ex} = \tau_{FNL}$ and $\tau_F = \tau_{SF}$), and added another worst-case scenario assumption, where all switching particles switch to an active state and do not have an FNL: $N_S = N_{SA} = N'_{SA}$.
After substitution, Equation 8 changes to:

$$\tau_{ARMD2} = N_A * \tau_{FNL} + N_A * \tau_F + N_S * \tau_{FNL} + N_S * \tau_F + N_S * \tau_{Ex}. \tag{9}$$

Since both N_S and τ_{Ex} are small, we may neglect their product. As a result, Algorithm 5 is more efficient than classical MD when $N_A + N_S \leq N$, which is generally true for ARMD.

The difference in the computation times of Algorithms 4 and 5 is:

$$\tau_{ARMD1} - \tau_{ARMD2} = (N - N_A - N_{SA}) * \tau_{FNL}. \tag{10}$$

If the sum of the number of active particles and the number of particles that switch to an active state is smaller than the total number of particles (which is true most of the time since the number of switching particles is small), then Algorithm 5 performs better than Algorithm 4.

5 Results and discussions

The algorithms introduced above were validated on the following benchmarks:

1. Systems of Lennard-Jones particles.
2. Simulation of Crystal NaCl.
3. High-velocity impact of nanodroplet.
4. Simulation of a polymer in the solvent.

We note that, since the ARMD methodology has been validated beforehand[29,31], the aim of these benchmarks is to validate the single-pass incremental force update algorithms.

5.1 Systems of Lennard-Jones particles

We performed simulations of different numbers (500, 4000 and 108000) of Lennard-Jones particles using the AR integrator in the NVE ensemble. All simulations were carried out in reduced units (lj units) using the LAMMPS MD package. Particles were generated on a fcc lattice with density 0.8442 and initial velocities were assigned according to the Boltzmann distribution at temperature $k_BT = 1.44$. For all simulations, we used a time-step of 0.005 and interactions beyond a distance of 2.5σ were ignored. Periodic boundary conditions were employed in all three directions.

We performed two series of benchmarks. In the first series, the percentage of restrained particles was constrained by assigning very high AR parameters to a specific number of particles (and zero AR values to the others), to ensure they would not switch their state and remain restrained. In the second series, we let particles switch their state and assigned all particles the same (lower) AR parameters.

We compared the run time of our algorithms with that of LAMMPS algorithms. The reference simulations were performed using LAMMPS neighbor lists and force algorithms, whereas ARMD simulations were performed using the ANLs and single-pass incremental force algorithms. The neighbor list was updated at every 20 time-steps. For speedup measurements, we computed the average time spent at each integration step and the time spent in the construction of the ANL as well. These values were then compared to the average time spent per integration step and per construction of neighbor list in the reference simulations.

Series 1 (constrained number of restrained particles): Figure 2 shows the achieved speedup in constructing the ANL vs. the percentage of restrained particles. The construction of both the ANL and the FNL required a $27r_c^3$ volume to be searched, in contrast to the HNL, which required half of that volume. As a result, the construction time for building the ANL or the FNL was twice that of the HNL. However, in order to have a speedup when constructing the ANL, at least 50% of the particles are required to be restrained. Also, we found that the obtained speedup in constructing the ANL was the same regardless of the number of particles present in the systems. In conclusion, we observed a 2X (resp. 4X) speedup in constructing the ANL while restraining 80% (resp. 90%) of the particles.

Figure 3 shows the achieved speedup per integration step with respect to the percentage of restrained particles in the system. Thanks to the new single-pass force update algorithm, there is no constraint on the minimum number of restrained particles. In fact, even with only 40% of particles restrained, we achieved a $1.5X$ speedup. With 60% of the particles restrained, our algorithm performed twice as fast as the LAMMPS algorithm. Furthermore, 3X to 5X speedups were observed when $80 - 90\%$ particles were restrained.

Finally, we measured the overall speedup which encompasses the time to built the NLs and the time to perform integration steps. Figure 4 shows the overall speedup vs. the percentage of restrained particles. For less than 50% of restrained particles, the new single-pass incremental algorithm is twice faster than the two-pass incremental algorithm. As expected, the single-pass incremental algorithm is always faster than the two-pass incremental algorithm due to the reduced number of force computations. When comparing with traditional MD and the number of restrained particles was less than 20%, however, no speedup was achieved. This is explained by the cost of constructing the ANL, which is approximately twice the cost of constructing the HNL. Overall, A similar trend to the speedup in Figure 3 was observed with the overall speedup: a 2X speedup was observed with 60% particles restrained and up to 5X speedup with 90% particles restrained (Figure fig:case1overallspeedup).

Series 2 (identical AR parameters for all particles): In this series of benchmarks, particles were allowed to switch states during simulation. Table 1 shows that the average number of switched particles at each time step was much smaller than the number of particles in the system. Thanks to this, switched particles did not have much influence on the obtained speedup, which reproduced the patterns observed in Series 1, and the single-pass force update algorithm always performed better than the two-pass incremental algorithm.

For the system containing 500 particles, with the single-pass incremental algorithm, a 2.1X to 3.5X speedup was observed with 81% to 86% particles restrained, and a maximum speedup of 4.5X was achieved when 92% of the particles were restrained. The AR parameters, the average number of switched particles and the average number of restrained particles can be found in Table 1.

For the system containing 4000 particles, a $3.8X$ speedup was attained when 91% of the

particles were restrained, whereas in Series 1 the corresponding speedup was $4.5X$ with the same number of restrained particles. When 96% and 98.5% of the particles were restrained, we achieved a 6X and 8.9X speedup respectively. A 1.2X speedup was observed with the 57% of the particles as restrained particles. In another system containing 32000 particles, a 2X speedup was achieved with 46% of restrained particles. A 4X to 6X speedup was observed with 85% to 98% of particles restrained. For the system containing 108000 particles, a maximum speedup of 7.5X was observed with 97% of restrained particles. The reduction in overall speedups in Series 2 as compared to Series 1 is due to switched particles (building ANL and computing force components), and the fact that the observed percentages of restrained particles are averages.

5.2 Simulation of NaCl

In order to validate our algorithm on a computationally expensive potential, we simulated a system containing 8000 (4000 Na^+ and 4000 Cl^-) NaCl particles. In this NaCl system, charged particles Na^+ and Cl^- interact via electrostatic interactions. In MD, during force calculations, most of the computational time is used in calculating electrostatic forces. In the NaCl system, around 90% of the total time is spent in the computation of electrostatic interactions.

The NaCl system was simulated using the Tosi-Fumi (TF) potential augmented with a coulombic potential. The TF potential is given by equation 11, and this potential is broadly used in simulation of alkali halides.

$$U(r) = A\ exp\left(\frac{\sigma - r}{\rho}\right) - \frac{C}{r^6} + \frac{D}{r^8} \qquad r < R_c \tag{11}$$

The first term in equation 11 is the Born-Mayer exponential repulsive term and the second term involves 8, 6 Van der Waals attractive interaction. Parameters for TF potential are taken from reference[34]. Instead of calculating electrostatic forces based on Ewald summation method, we used Wolf summation. The Wolf method is computationally efficient ($O(N)$) as compared to the Ewald-based methods. Furthermore, the Wolf method can use neighbor

lists for computing electrostatic forces:

$$E_{Wolf} = \frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j\neq i \\ r_{ij}<R_c}}\left[\left(\frac{q_i q_j \operatorname{erfc}(\alpha r_{ij})}{r_{ij}} - \frac{q_i q_j \operatorname{erfc}(\alpha R_c)}{R_c}\right)\right] - \left(\frac{\operatorname{erfc}(\alpha R_c)}{2R_c} + \frac{\alpha}{\sqrt{\pi}}\right)\sum_{i=1}^{N}\frac{q_i^2}{4\pi\epsilon_0} \tag{12}$$

In equation 12, erfc is the complementary error function, q_i and q_j represent the point charges on particles i and j, α is the damping parameter and R_c is the cut-off radius. Details regarding the Wolf method can be found from references[23–28]. For the Wolf method, we used the same parameters as those mentioned in the literature[35].

Cut-off distances of 7.5Å and 15Å were used for the TF potential and Wolf summation respectively. The system was simulated for 100000 time-steps using an integration time-step of $2fs$ in the NVE ensemble. In order to measure the speedup, we performed reference MD simulations as well as ARMD simulations with different AR parameters. For timing measurements, we ran 50 independent simulations (MD and different ARMD simulations) for each combination of parameters and timings were averaged over these 50 simulations.

Different AR parameters (Table 2) give rise to different average numbers of active and switched particles. Figure 6 shows the speedup obtained with ARMD compared to MD. We achieved a 2X speedup with 65% of restrained particles, and a 4X to 10X speedup was achieved with 82% to 96% of restrained particles. This benchmark shows that combining the Wolf method with ARMD significantly reduces the number of calculations, which may result in a significant speedup.

5.3 High-velocity impact of a nanodroplet

The high-velocity impact of a nanodroplet on a crystal surface changes the state of the crystal to an amorphous state, and this process is known as amorphization[36,37]. We performed this benchmark to show that ARMD simulations may offer important speedups on such processes. Our model system contains three types of particles, namely 1) nanodroplet, 2) target slab and 3) boundary particles. The nanodroplet consists of 2891 identical particles that are spherically distributed in a hexagonal close-packed arrangement (blue particles shown in Figure 7); the target slab consist of 344,988 identical atoms on the fcc lattice (grey and red

particles shown in Figure 7); the boundary of the slab remains fixed and has no velocity (green particles shown in Figure 7). Interactions among particles were computed using the Lennard-jones potential. All simulations were performed using a 1 fs integration time step and ran for 75 ps. The initial velocity of the nanodroplet was set to 4 km/s (in negative z direction). Due to the high velocity of the nanodroplet, the neighbor list was constructed every fifth time step.

In order to measure the amorphization process, we observed the radial distribution function (RDF) of the impact volume of the target slab (red particles in Figure 7) for different sets of AR parameters, and compared the obtained RDFs with the one obtained with a reference MD simulation. In this benchmark, most slab particles were initially restrained, and gradually started to switch to an active state after impact.

As shown in Figure 7, the crystalline structure at the impact area completely changes to a non-crystalline structure (amorphous state). This figure also shows that ARMD simulations allow us to trade between speed and accuracy. Lower values of AR parameters produce a trajectory similar to the MD trajectory, but higher values of AR parameters produce higher speedups. Figure 8 shows the obtained RDFs of the impact area (particles denoted as red in figure 7) at different time-steps. At the beginning of the simulation (at $t = 0ps$), the impact area had a crystalline structure. As the simulation proceeds, this crystalline structure started to deform and changed to an amorphous state (RDFs at $t = 15, 30, 45, 60$ and $75ps$). During the amorphization process, the RDFs obtained from ARMD simulations mostly coincide with the RDF obtained using MD (Figure 8). The RDFs of the impact area at $75ps$ obtained using both ARMD and MD had their first peak at 2.35Å and second peak at 4.5Å. In order to measure the speed obtained by ARMD, we ran each simulation 50 times and computed the average time spent. We measured the speedup with respect to the reference MD simulation. ARMD achieved 2.3-7X speedup with $56 - 85\%$ restrained particles as compared to MD. Table 3 shows the speedup obtained with different AR parameters. This benchmark shows that ARMD saves wall-clock time while obtaining the structural properties of a specific part of the system much faster than classical MD, thanks to the automatic particle state switching resulting from the AR Hamiltonian[29].

5.4 A single polymer chain in solution

This model is used to study the properties of a polymer in solution. This model is also a representative model of the biological coarse-grained model of protein in water. In this benchmark, we test the possibility for ARMD to obtain statistical averages faster than with MD. We performed ARMD of the system with different AR parameters and compared our results with those obtained using MD. A polymer chain of 30 monomers immersed in 4000 solvent particles was simulated in the NVT ensemble at temperature $k_BT = 1.2$ with density $\rho = 0.86$, while using a time step of 0.001. Periodic boundary conditions were imposed in all the three directions. The FENE potential was used for the backbone (eq. 13) and truncated Lennard-jones (WCA) potential was used for pair interactions. Initial velocities were assigned according to the Boltzmann distribution.

$$U_{FENE}(r) = -\frac{K}{2}R_0^2 \, ln\left(1 - \frac{r^2}{R_0^2}\right) \tag{13}$$

The system was initially minimized and then equilibrated for 10000 steps. After the equilibration period, the system was simulated for 10^9 steps in the NVT ensemble. Initial velocities were assigned using Maxwell-Boltzmann distribution at temperature $k_BT = 1.2$.

One of the main goal of MD is to compute the statistical properties of the system by calculating averages. Averages obtained by MD are time averages and, if simulation is long enough to be converged, these time averages are equal to ensemble averages (ergodic hypothesis). However, averages over a trajectory are subject to two types of errors: systematic bias and statistical errors. Systematic bias is due to the use of a discrete time step, and statistical errors occur due to the quality of sampling (and may be large if averages are obtained from an undersampled trajectory).

Statistical errors in time averages may be estimated by measuring the variance of an observable l. The variance can be measured either by correlation time analysis or by a block-averaging scheme[38–41]. In previous studies of ARMD, time correlation analysis was used to measure errors and variance[33], and we use the same approach for error estimations of ARMD trajectories in the present paper. The correlation time τ_l of an observable l is the simulation time required for a trajectory to de-correlate its value from an initial value l_0. Therefore, the correlation time for an observable provides an estimation of N_l^{ind}, the

number of statistically independent values of the observable present in the trajectory: if t_{sim} denotes the simulation length of a trajectory, then $N_l^{ind} \sim t_{sim}/\tau_l$. Larger values of N_l^{ind} suggests good sampling for the given observable. The time τ_l depends upon the nature of observable[42].

In ARMD, restraining positions of particles introduces additional correlation in the system, thus yielding a larger correlation time for a given observable, as compared to MD ($\tau_l^{ARMD} \geq \tau_l^{MD}$). If ARMD simulations have the same length as MD simulations, then statistical errors are larger:

$$\tau_l^{ARMD} \geq \tau_l^{MD} \text{ and } t_{sim}^{ARMD} = t_{sim}^{MD} \implies N_{l_{ARMD}}^{ind} \leq N_{l_{MD}}^{ind}$$

Fortunately, statistical errors of averages obtained by ARMD may be reduced, and more statistically independent values may be obtained, by performing simulations with longer lengths (more time steps), since each time step costs less in wall-clock time. Therefore, the overall speedup in the NVT ensemble is a function of the speedup obtained in wall-clock time at each time step, and the time required to attain a given precision in the estimation of a given observable[33]. This speedup can be expressed as:

$$S = S_{algo}\frac{\sigma_{MD}^2}{\sigma_{ARMD}^2} \tag{14}$$

where S_{algo} represents the computational speedup at each time step adaptive algorithms (ANLs and single-pass incremental force update algorithms), σ_{MD}^2 is the variance of a given observable when using MD, and σ_{ARMD}^2 is the variance of the same observable when using ARMD.

In this benchmark of a polymer in solvent, we chose the end-to-end distance of the polymer as an observable, and computed the correlation times with different AR parameters. Figure 10 shows the time correlation functions computed for end-to-end distances. As expected, the correlation function in the MD case is reduced in fewer time steps compared to the correlation functions in ARMD simulations. In wall-clock time, however, some ARMD trajectories decorrelate the end-to-end distance faster than MD, due to the reduction in the average cost of a time step.

Table 4 shows the averaged values of end-to-end distances and radius of gyration of the polymer chain. The averages obtained with MD and ARMD are approximately the same.

This experimentally shows how, when the observable is converged, averages obtained with ARMD are the same as the ones obtained with MD (see[29] for a mathematical proof). Figure 11 shows the 2D projection of all trajectories as a function of end-to-end distance and radius of gyration of the polymer. Here as well we obtain unbiased position-dependent averages. This illustrates that ARMD samples the same conformational space as the MD in the NVT ensemble[29].

6 Conclusions

We have presented a novel single-pass force update algorithm to speed up ARMD simulations. Unlike our previous two-pass algorithms, the new algorithm may result in a speedup even when a small percentage of particles is restrained. We have validated the approach on several benchmarks, and have shown that the single-pass algorithm may be applied to computing electrostatic interactions with the Wolf method. We showed how ARMD may be used to converge a given position-dependent observable faster than with MD for systems where precise information is needed at a specific part of the system only. On the other hand, ARMD methodology might not be useful to study dynamical properties.

All algorithms have been implemented as a modified version of LAAMPS, which will be made available through SAMSON Connect (`https://www.samson-connect.net`). In the future, we would like to investigate the development of other incremental algorithms for long-range interactions, and apply the AR methodology to diverse systems which might benefit from restraining some particles, including ion channels, enzymatic reactions, defect and crack propagation, etc.

6.1 Acknowledgments

We gratefully acknowledge funding from Rhone-Alpes Region through the ARC program, and the European Research Council through the ERC Starting Grant n. 307629.

References

1. R. O. Dror, R. M. Dirks, J. P. Grossman, H. Xu, and D. E. Shaw, Annual Review of Biophysics **41**, 429 (2012), 00390 bibtex: dror_biomolecular_2012.

2. S. A. Adcock and J. A. McCammon, Chemical Reviews **106**, 1589 (2006), ISSN 0009-2665, 00659.

3. M. Karplus and J. A. McCammon, Nature Structural & Molecular Biology **9**, 646 (2002), ISSN 1072-8368, 01851 bibtex: karplus_molecular_2002.

4. W. F. van Gunsteren and H. J. C. Berendsen, Angewandte Chemie International Edition in English **29**, 992 (1990), ISSN 1521-3773, 01454.

5. A. Amadei, A. B. M. Linssen, and H. J. C. Berendsen, Proteins: Structure, Function, and Bioinformatics **17**, 412 (1993), ISSN 1097-0134, 02111.

6. R. Elber and M. Karplus, Science **235**, 318 (1987), ISSN 0036-8075, 1095-9203, 00697.

7. M. Levitt, Journal of Molecular Biology **168**, 595 (1983), ISSN 0022-2836, 00400.

8. A. Laio and M. Parrinello, Proceedings of the National Academy of Sciences **99**, 12562 (2002), ISSN 0027-8424, 1091-6490, 02437 bibtex: laio_escaping_2002.

9. D. Hamelberg, J. Mongan, and J. A. McCammon, The Journal of Chemical Physics **120**, 11919 (2004), ISSN 0021-9606, 1089-7690, 00662.

10. A. Barducci, G. Bussi, and M. Parrinello, Physical Review Letters **100**, 020603 (2008), 00608.

11. A. Barducci, M. Bonomi, and M. Parrinello, Wiley Interdisciplinary Reviews: Computational Molecular Science **1**, 826 (2011), ISSN 1759-0884, 00608.

12. C. Clementi, Current Opinion in Structural Biology **18**, 10 (2008), ISSN 0959-440X, 00247.

13. P. J. Bond, J. Holyoake, A. Ivetac, S. Khalid, and M. S. P. Sansom, Journal of Structural Biology **157**, 593 (2007), ISSN 1047-8477, 00256.

14. V. Tozzini, Current Opinion in Structural Biology **15**, 144 (2005), ISSN 0959-440X, 00694.

15. S. Riniker and W. F. v. Gunsteren, The Journal of Chemical Physics **137**, 044120 (2012), ISSN 0021-9606, 1089-7690, 00020 bibtex: riniker_mixing_2012.

16. C. H. Bennett, Journal of Computational Physics **19**, 267 (1975), ISSN 0021-9991, 00080.

17. C. W. Hopkins, S. Le Grand, R. C. Walker, and A. E. Roitberg, Journal of Chemical Theory and Computation **11**, 1864 (2015), ISSN 1549-9618, 00031.

18. D. E. Shaw, M. M. Deneroff, R. O. Dror, J. S. Kuskin, R. H. Larson, J. K. Salmon, C. Young, B. Batson, K. J. Bowers, J. C. Chao, et al., Commun. ACM **51**, 91 (2008), ISSN 0001-0782, 00419.

19. B. Garzn, S. Lago, and C. Vega, Chemical Physics Letters **231**, 366 (1994), ISSN 0009-2614, 00039.

20. I. G. Tironi, R. Sperb, P. E. Smith, and W. F. van Gunsteren, The Journal of chemical physics **102**, 5451 (1995).

21. M. Poursina and K. S. Anderson, Journal of Computational Physics **270**, 613 (2014), ISSN 0021-9991, 00001.

22. T. Darden, D. York, and L. Pedersen, The Journal of Chemical Physics **98**, 10089 (1993), ISSN 0021-9606, 1089-7690, 11098.

23. D. Wolf, P. Keblinski, S. R. Phillpot, and J. Eggebrecht, The Journal of Chemical Physics **110**, 8254 (1999), ISSN 0021-9606, 1089-7690, 00716.

24. C. J. Fennell and J. D. Gezelter, The Journal of Chemical Physics **124**, 234104 (2006), ISSN 0021-9606, 1089-7690, 00331.

25. Y. Yonezawa, I. Fukuda, N. Kamiya, H. Shimoyama, and H. Nakamura, Journal of Chemical Theory and Computation **7**, 1484 (2011), ISSN 1549-9618, 00019.

26. J. S. Hansen, T. B. Schrder, and J. C. Dyre, The Journal of Physical Chemistry B **116**, 5738 (2012), ISSN 1520-6106, 00038.

27. B. W. McCann and O. Acevedo, Journal of Chemical Theory and Computation **9**, 944 (2013), ISSN 1549-9618, 00014.

28. J. A. Baker and J. D. Hirst, Faraday Discuss. **169**, 343 (2014), ISSN 1359-6640, 1364-5498, 00000.

29. S. Artemova and S. Redon, Physical Review Letters **109**, 190201 (2012), 00013 bibtex: artemova_adaptively_2012.

30. M. Bosson, S. Grudinin, X. Bouju, and S. Redon, Journal of Computational Physics **231**, 2581 (2012), ISSN 0021-9991, 00017 bibtex: bosson_interactive_2012.

31. Z. Trstanova and S. Redon, Journal of Computational Physics **336**, 412 (2017).

32. K. K. Singh and S. Redon, Modelling and Simulation in Materials Science and Engineering (2017).

33. S. Redon, G. Stoltz, and Z. Trstanova, Journal of Statistical Physics pp. 1–37 (2016), ISSN 0022-4715, 1572-9613, 00005.

34. J. L. Aragones, E. Sanz, C. Valeriani, and C. Vega, The Journal of Chemical Physics **137**, 104507 (2012), ISSN 0021-9606, 1089-7690, 00013.

35. G. S. Fanourgakis, The Journal of Physical Chemistry B **119**, 1974 (2015), ISSN 1520-6106, 00005.

36. F. Saiz and M. Gamero-Castao, Journal of Applied Physics **112**, 054302 (2012), ISSN 0021-8979, 1089-7550, 00009.

37. F. Saiz and M. Gamero-Castao, AIP Advances **6**, 065319 (2016), ISSN 2158-3226, 00001.

38. E. Lyman and D. M. Zuckerman, Biophysical Journal **91**, 164 (2006), ISSN 0006-3495, 00087.

39. A. Grossfield and D. M. Zuckerman, Annual reports in computational chemistry **5**, 23 (2009), ISSN 1574-1400, 00124.

40. D. M. Zuckerman, Annual review of biophysics **40**, 41 (2011), ISSN 1936-122X, 00000.

41. E. Lyman and D. M. Zuckerman, The journal of physical chemistry. B **111**, 12876 (2007), ISSN 1520-6106, 00047.

42. T. D. Romo and A. Grossfield, Biophysical journal **106**, 1553 (2014).

Figure 1: Different types of force components present in the system. Green particles are active and blue particles are restrained. The force components F_{AA}, F_{AR} and F_{RR} represent forces acting between active particles, forces between active and restrained particles, and forces between restrained particles.

Figure 2: Speedup when constructing the ANL (compared to the HNL) as a function of the percentage of restrained particles.

Figure 3: Series 1: Speedup per integration step.

Figure 4: Series 1: Overall speedup obtained as a function of the percentage of restrained particles using the single-pass and two-pass incremental algorithms. The new single-pass algorithm always performs better than the two-pass algorithm.

Figure 5: Series 2: Speedup achieved with the single-pass and two-pass incremental algorithms over MD as a function of the percentage of restrained particles.

Figure 6: Speedup achieved with ARMD for ANL construction and per time step in the NaCl benchmark.

Figure 7: Cross-section view of the nanodroplet impact at 0, 15, 30, 45, 60 and 75 ps. Red particles belong to the impact area. Green particles represent the fixed boundary of the impact slab.

Figure 8: Comparison of the RDFs of the impact area obtained with MD and ARMD. The RDFs obtained by ARMD with different AR parameters at different time-steps mostly coincide with the RDFs obtained by MD.

Figure 9: Instantaneous temperature of the system with different AR parameters.

Figure 10: Time correlation function of the end-to-end distance of the polymer. The left part shows that MD takes fewer iterations to decorrelate the end-to-end distance when compared to ARMD. The right part shows that, in wall-clock time, however, some ARMD simulations converge up to twice faster than a MD simulation.

Figure 11: Projection of trajectories on the end-to-end distance (R) and radius of gyration (R_g) of the polymer (colors represent the number of conformations in each bin). This figure illustrates that ARMD simulations produce unbiased positional statistics.

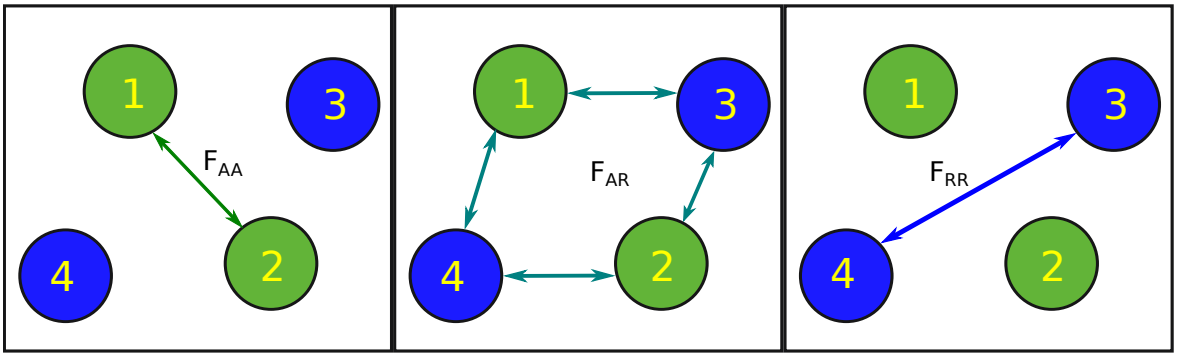


Figure 1
Krishna Kant Singh and Stephane Redon
J. Comput. Chem.

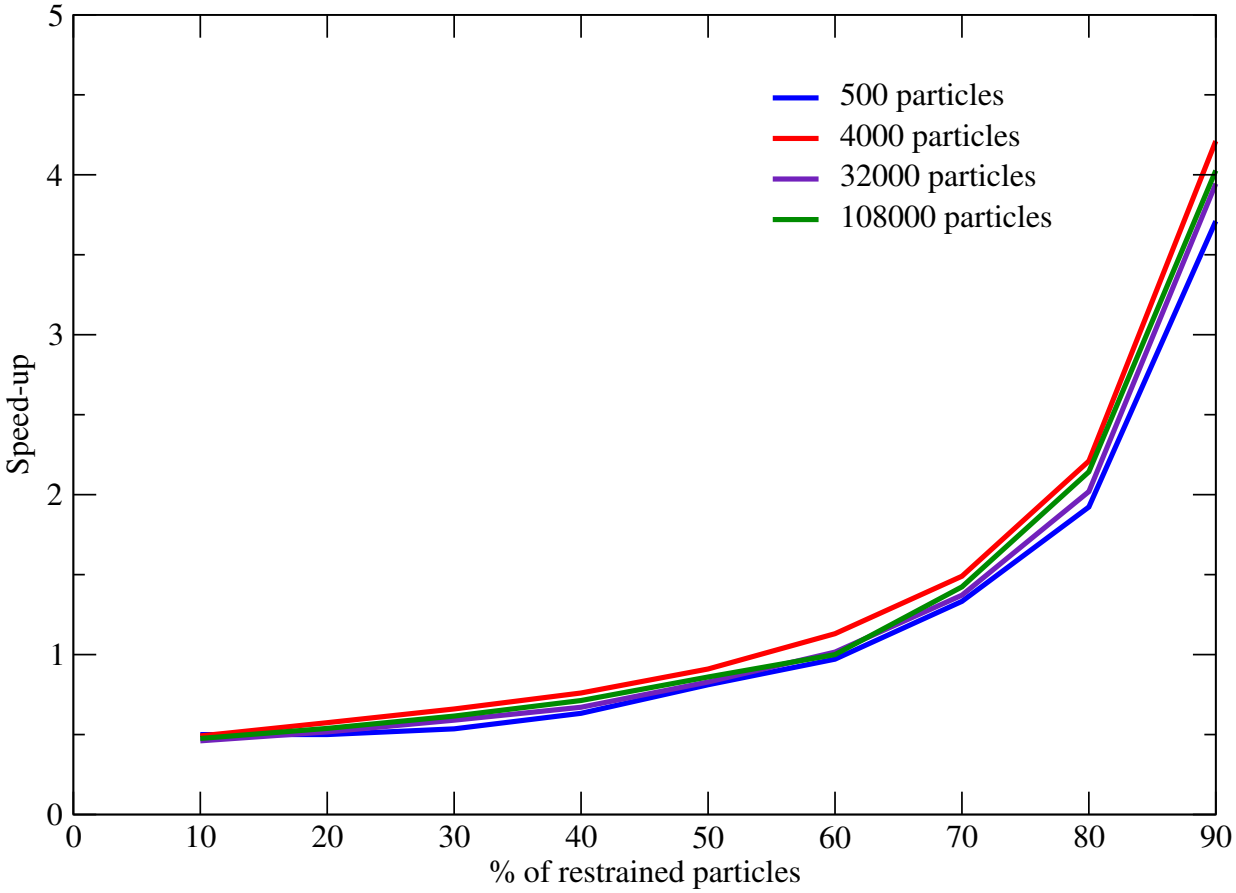


Figure 2
Krishna Kant Singh and Stephane Redon
J. Comput. Chem.

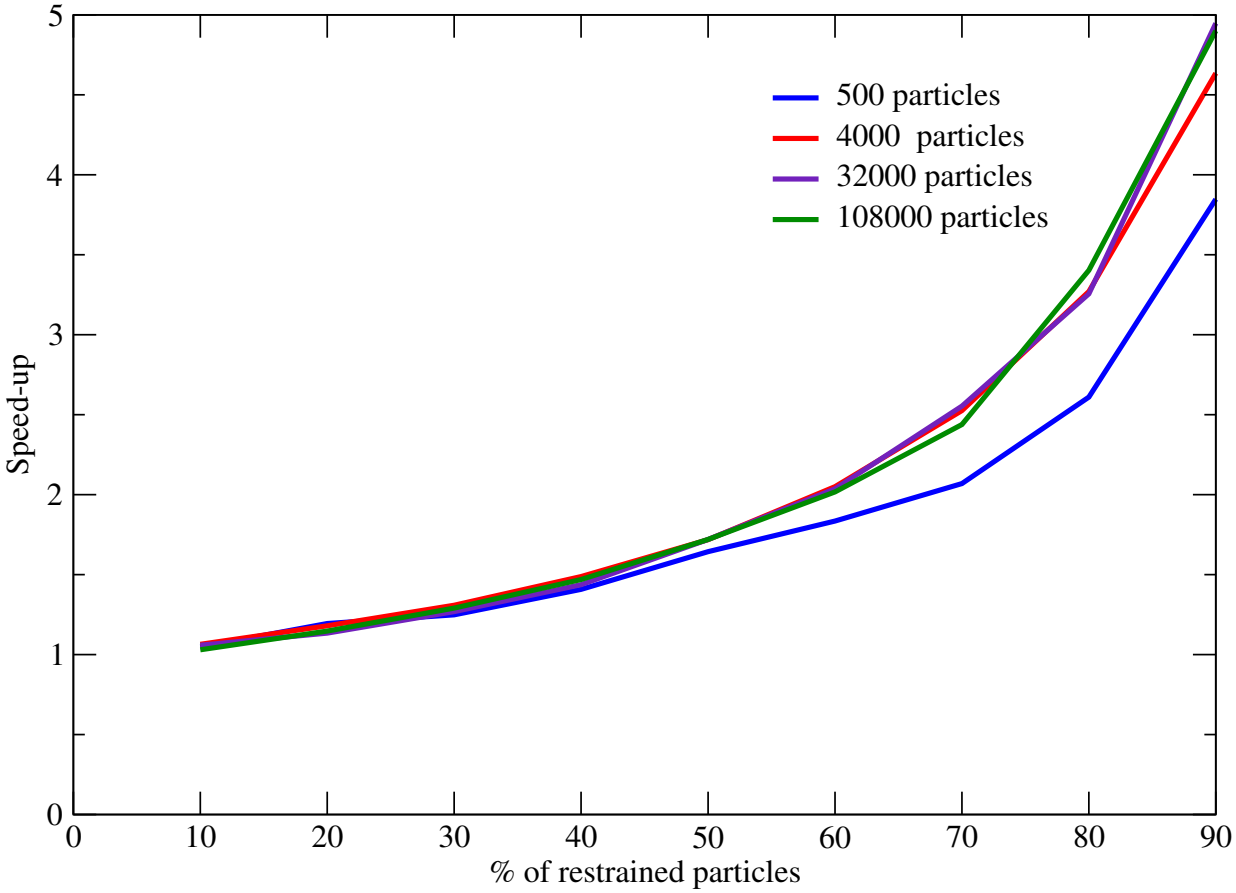


Figure 3
Krishna Kant Singh and Stephane Redon
J. Comput. Chem.

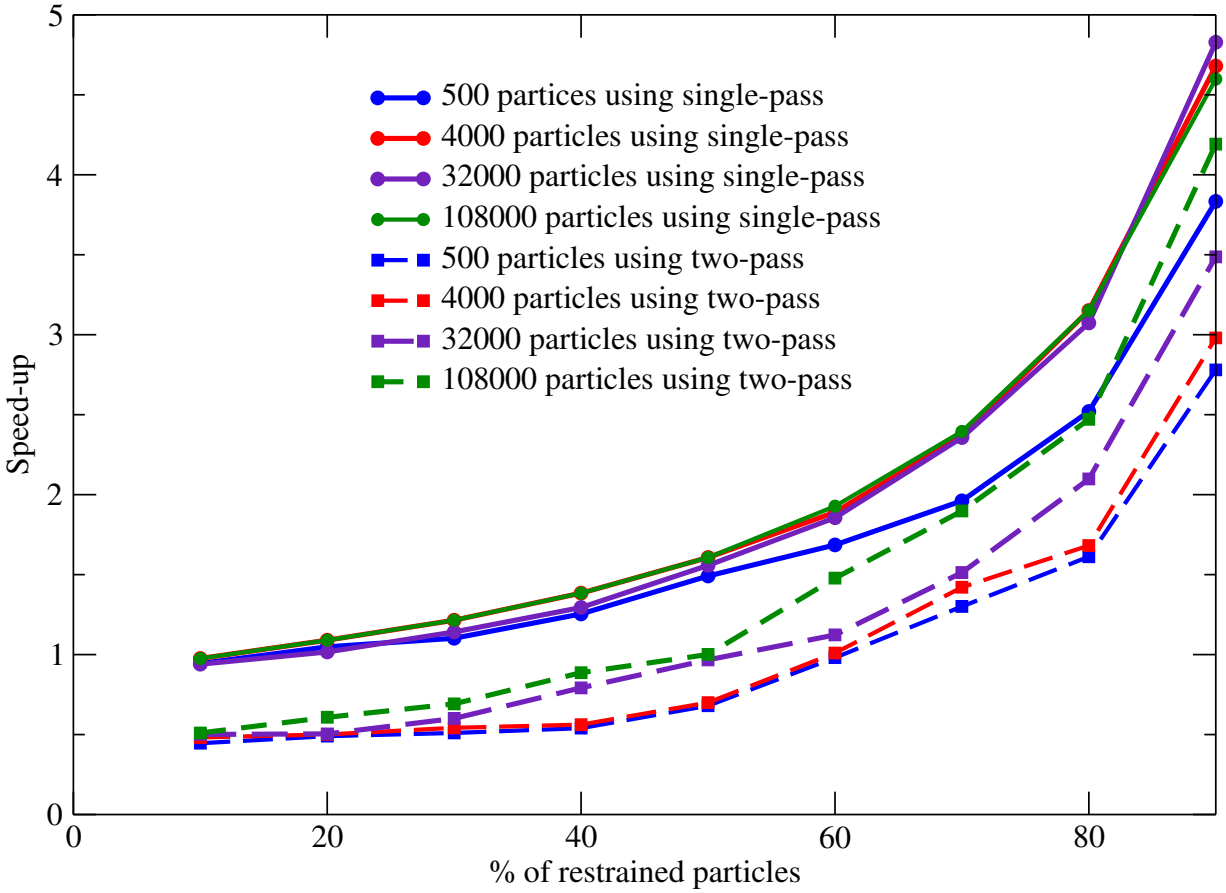


Figure 4
Krishna Kant Singh and Stephane Redon
J. Comput. Chem.

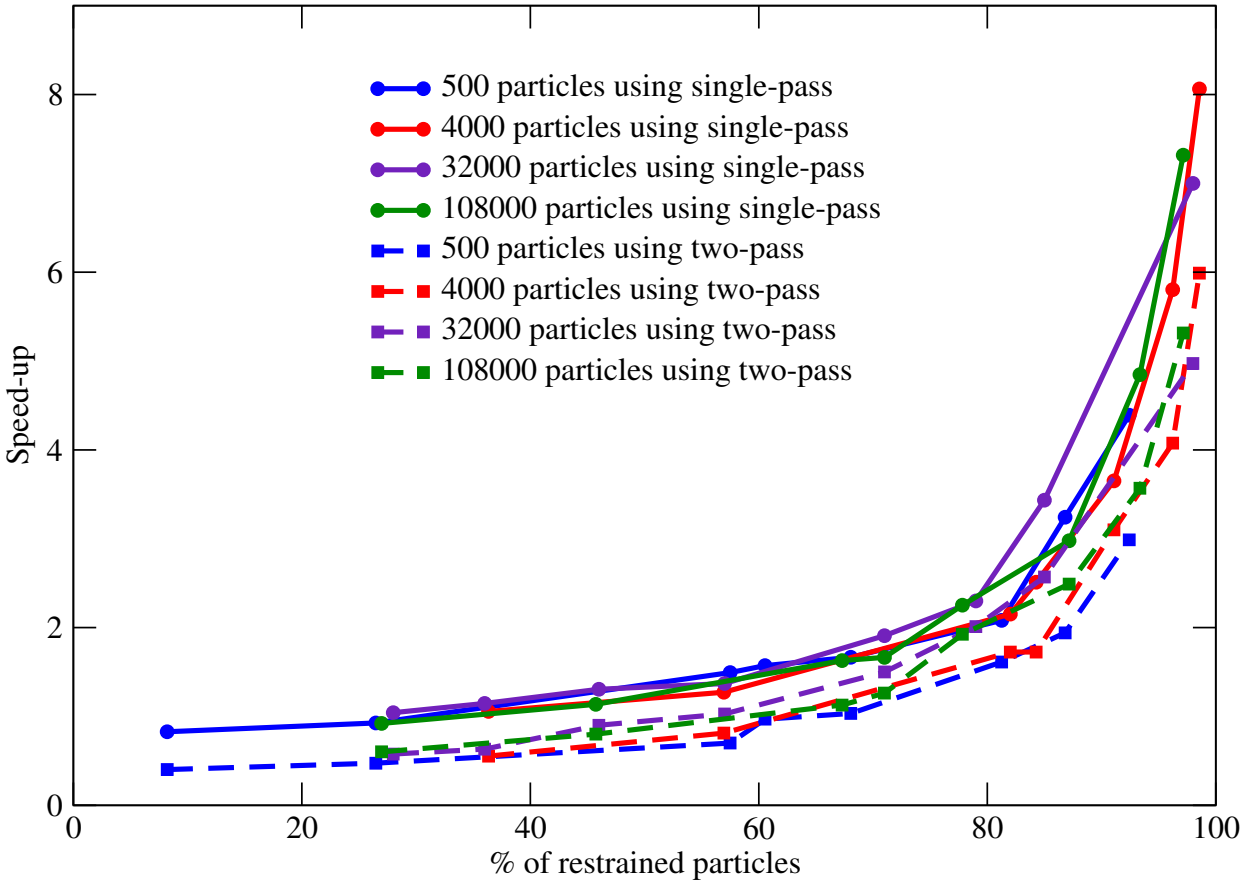


Figure 5
Krishna Kant Singh and Stephane Redon
J. Comput. Chem.

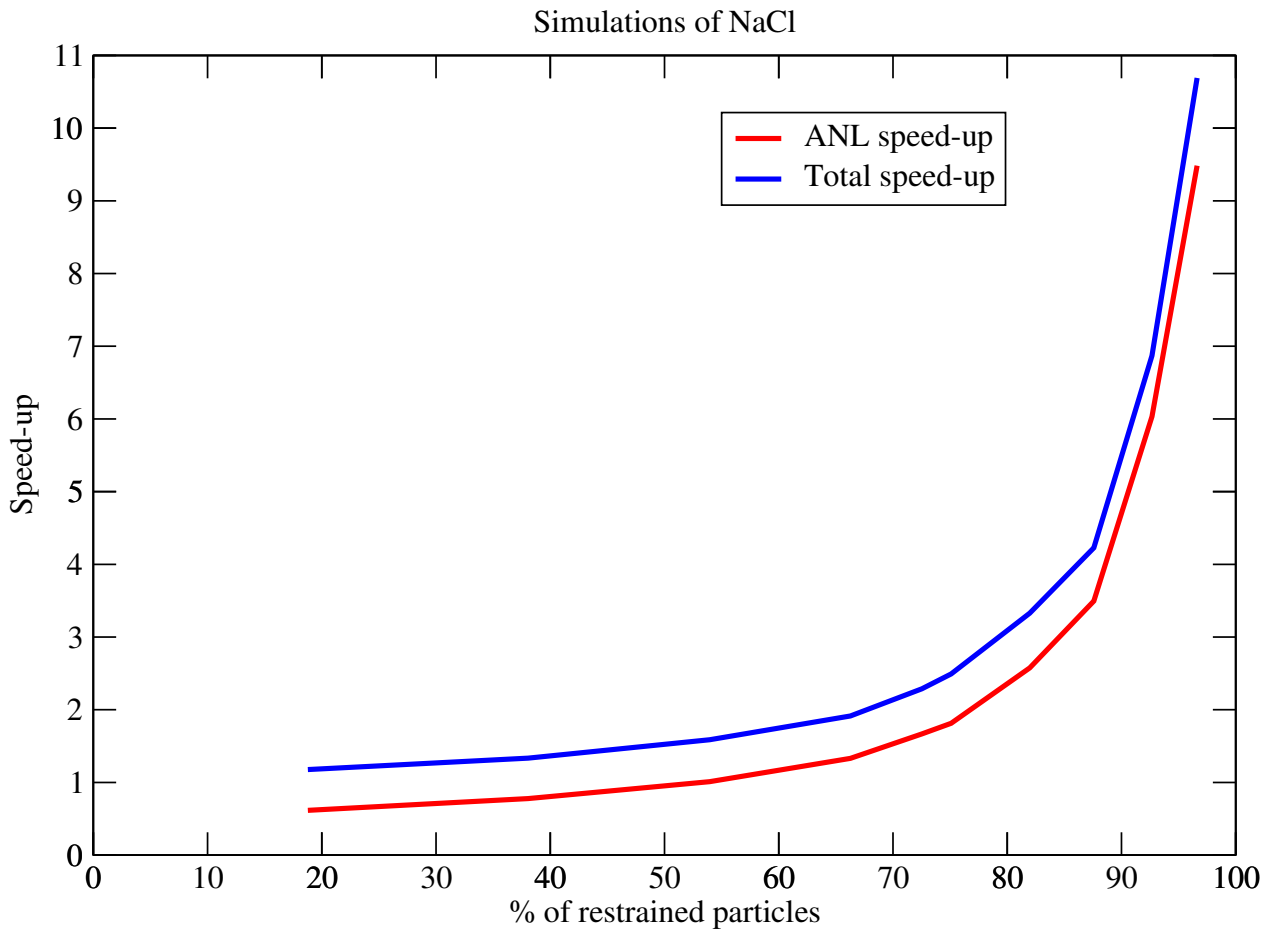


Figure 6
Krishna Kant Singh and
Stephane Redon
J. Comput. Chem.

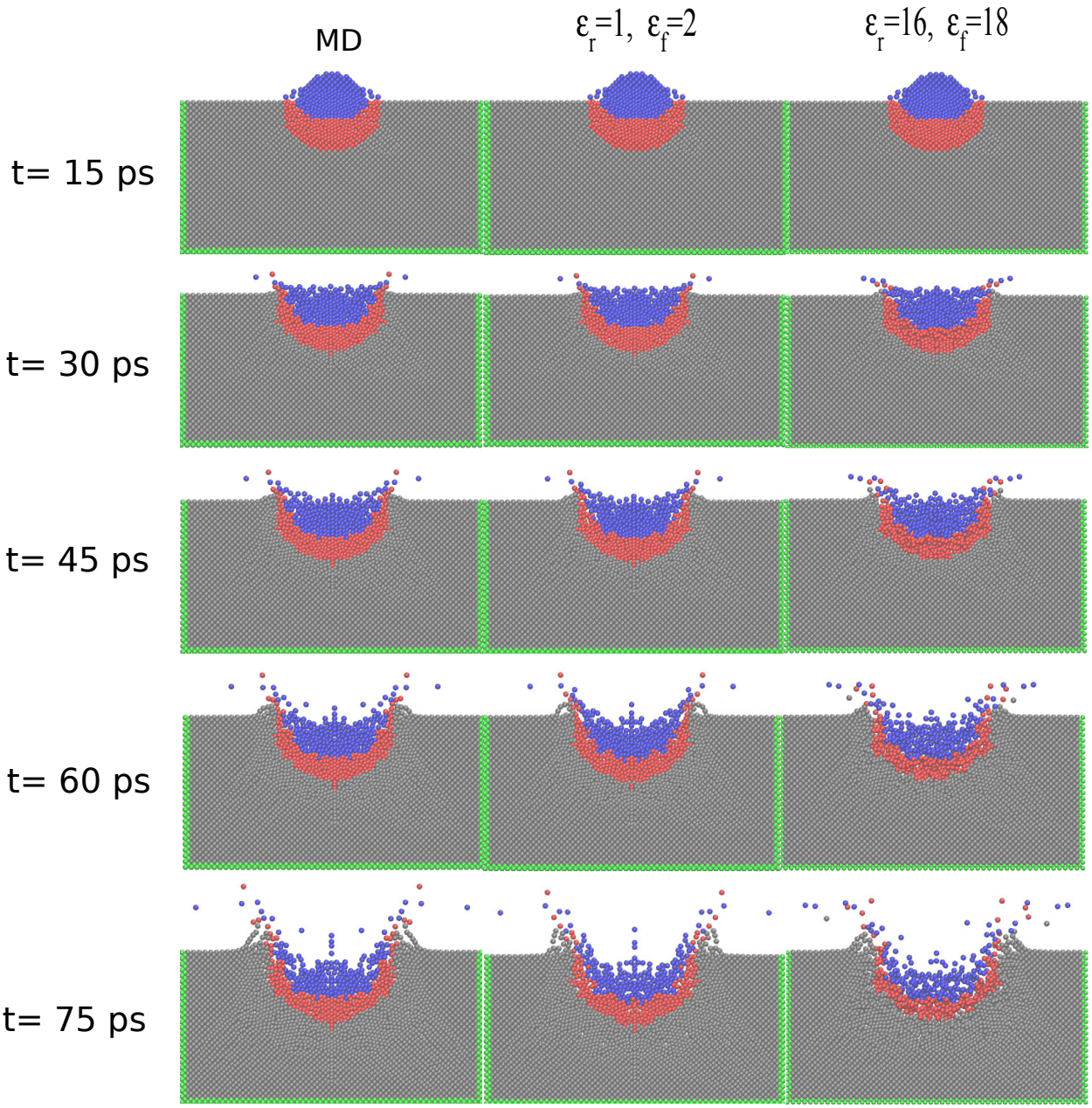


Figure 7
Krishna Kant Singh and Stephane Redon
J. Comput. Chem.

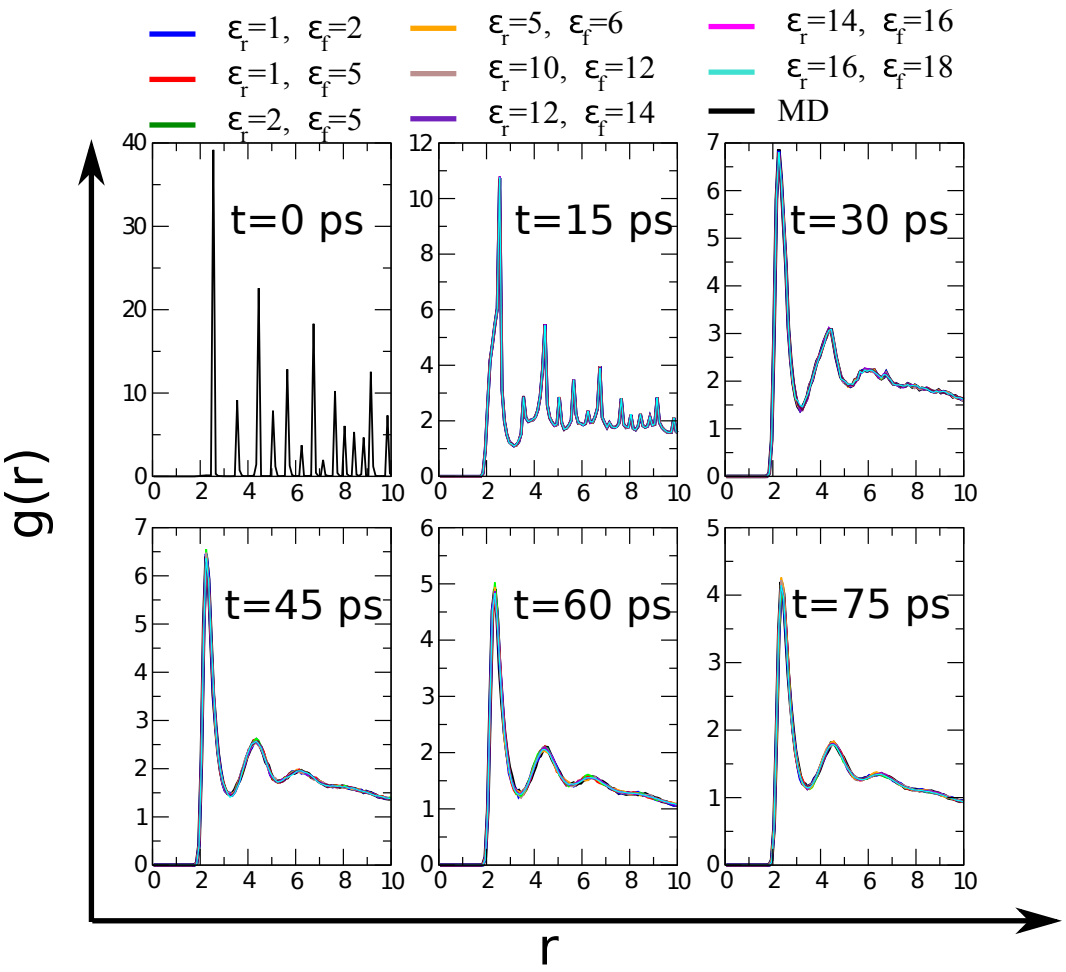


Figure 8
Krishna Kant Singh and Stephane Redon
J. Comput. Chem.

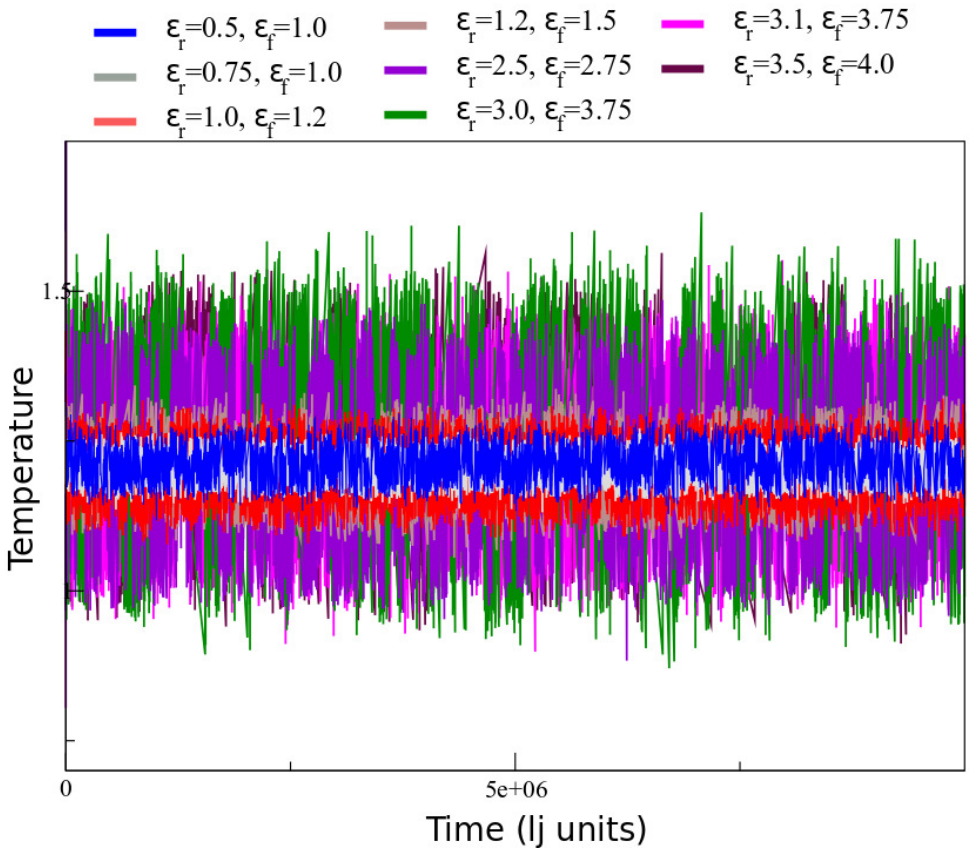


Figure 9
Krishna Kant Singh and Stephane Redon
J. Comput. Chem.

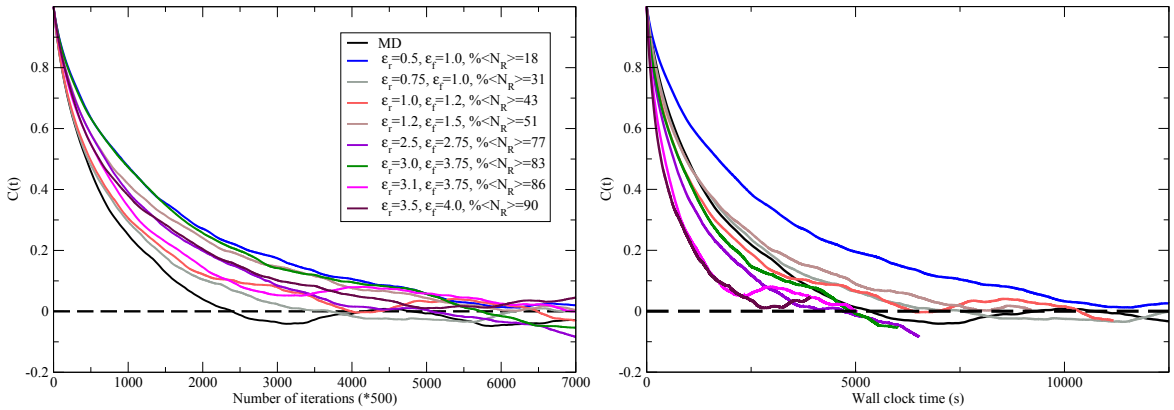


Figure 10
Krishna Kant Singh and Stephane Redon
J. Comput. Chem.

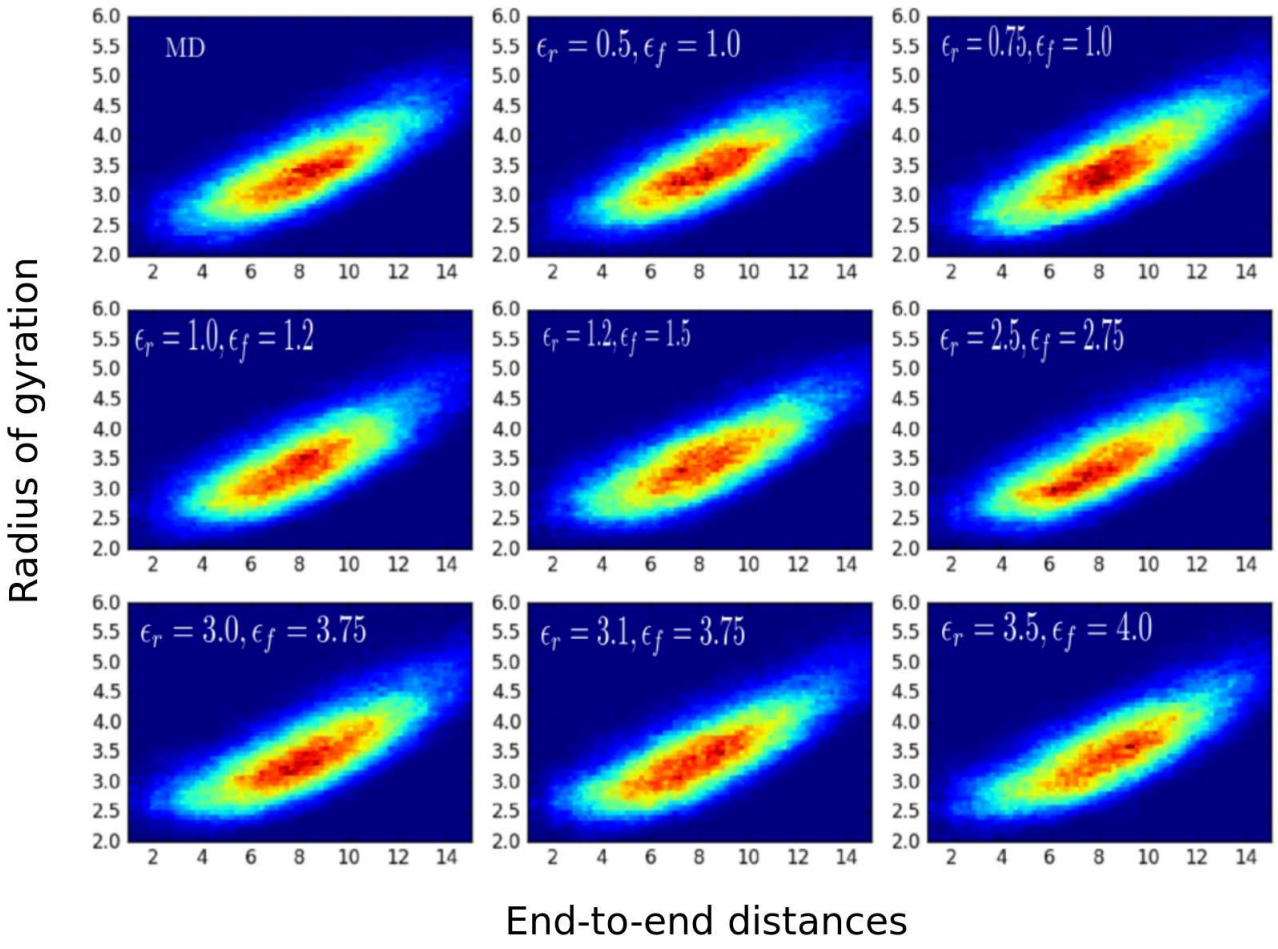


Figure 11

Krishna Kant Singh and

Stephane Redon

J. Comput. Chem.

ϵ_r	ϵ_f	500		4000		32000		108000	
		$\% < N_R >$	$\% < N_S >$	$\% < N_R >$	$\% < N_S >$	$\% < N_R >$	$\% < N_S >$	$\% < N_R >$	$\% < N_S >$
0.5	1	8.21	0.015	36.34	0.0075	28.12	0.0011	26.97	0.0054
1	2	26.47	0.024	56.94	0.0095	36.53	0.0091	45.73	0.0041
1.5	2	57.48	0.034	82.015	0.0042	46.96	0.0084	67.99	0.0073
2	4	60.54	0.029	84.27	0.0024	57.24	0.0021	70.99	0.0011
2.5	4	68.04	0.025	87.23	0.0092	71.32	0.0062	77.81	0.0063
3	5	81.25	0.021	91.08	0.0055	79.14	0.0055	87.16	0.0077
3.5	5	86.81	0.018	96.23	0.0032	85.75	0.0031	93.36	0.0021
4	5	92.42	0.014	96.23	0.0085	98.34	0.0022	97.15	0.0015

Table 1: AR parameters for Lennard-Jones systems. The average percentage of particles that switch states at each time step is very small, and does not significantly affect performance.

ϵ_r	ϵ_f	$\% < N_R >$	$\% < N_S >$
0.1	1	18.78	0.0032
0.5	1	38.056	0.010
1	2	53.95175	0.0087
1.5	2	66.272125	0.0063
2	5	72.53125	0.0025
2.5	5	75.0875	0.0082
3	5	81.966625	0.0042
3.5	5	87.5818625	0.0072
4	5	92.67155	0.0065
4.5	5	96.6208125	0.0054

Table 2: AR parameters used in the simulation of the $NaCl$ system.

ϵ_r	ϵ_f	$\% < N_R >$	speedup
1	2	56.4	2.3
1	5	60.79	2.9
2	5	61.87	3.1
5	6	66.49	4.25
10	12	72.49	4.7
12	14	75.53	5.1
14	16	79.34	5.9
16	18	84.21	6.82

Table 3: AR parameters used for performing ARMD simulations of the high-velocity impact of the nanodroplet.

	MD	ϵ_r/ϵ_f							
		.5/1	.75/1.0	1.0/1.2	1.2/1.5	2.5/2.75	3.0/3.75	3.1/3.75	3.5/4.0
$\% < N_R >$	0	18	31	43	51	77.8	83	86	90
$R =< R^2 >^{\frac{1}{2}}$	8.4404	8.481	8.634	8.3013	8.368	8.394	8.569	8.3919	8.556
$R_G =< R_G^2 >^{\frac{1}{2}}$	3.523	3.547	3.585	3.520	3.55	3.535	3.546	3.534	3.569

Table 4: Summary of statistical properties obtained by MD and ARMD for the polymer benchmark. R is the end-to-end distance of the chain and R_G is the radius of gyration. Statistical averages obtained from ARMD and MD are similar.